Sampling Chaotic Trajectories Quickly in Parallel

J. Machta[*]

Department of Physics, University of Massachusetts, Amherst, MA 01003-3720

(Dated: February 8, 2008)

Abstract

The parallel computational complexity of the quadratic map is studied. A parallel algorithm is described that generates typical pseudotrajectories of length t in a time that scales as $\log t$ and increases slowly in the accuracy demanded of the pseudotrajectory. Long pseudotrajectories are created in parallel by putting together many short pseudotrajectories; Monte Carlo procedures are used to eliminate the discontinuities between these short pseudotrajectories and then suitably randomize the resulting long pseudotrajectory. Numerical simulations are presented that show the scaling properties of the parallel algorithm. The existence of the fast parallel algorithm provides a way to formalize the intuitive notion that chaotic systems do not generate complex histories.

[*]machta@physics.umass.edu

I. INTRODUCTION

Chaotic systems cannot be predicted for very long times because of the exponential divergence of nearby trajectories. Associated with the divergence of trajectories is a lack of history dependence; the current behavior of the system is not dependent on the past behavior. The absence of history dependence can be understood in various ways. Here I take a computational perspective on chaotic systems and analyze trajectories in terms of computational complexity. If, using a massively parallel computer, a typical long trajectory can be manufactured in far fewer parallel steps than the actual length of the trajectory then the trajectory lacks history dependence or historical complexity. Alternatively, if parallelism does not allow one to generate a typical trajectory much more quickly than its actual length then the trajectory displays a complex history dependence.

These considerations are illustrated using the one-dimensional quadratic map,

$$x_{n+1} = rx_n(1 - x_n) \equiv f(x_n) \tag{1}$$

with $x_n \in [0, 1]$ and $0 < r < 4$. The objective is to produce typical trajectories for the map but since computing devices are necessarily restricted to finite precision we are really interested in generating typical *pseudotrajectories* [1]. A pseudotrajectory of accuracy δ is a sequence $\{y_n | n = 0, \cdots, t\}$ such that for all n $(0 \le n < t)$,

$$|y_{n+1} - f(y_n)| < \delta. \tag{2}$$

We suppose that the parallel computer works to a precision substantially better than δ so that bounds such as given in Eq. 2 can be checked with reasonable certainty.

For chaotic dynamics with a positive Lyapunov exponent λ, a pseudotrajectory and an exact trajectory that are initially equal remain close only for a time that is roughly given by $(\log \delta)/\lambda$. Nonetheless, for small δ a pseudotrajectory will have nearly the same statistical properties as a real trajectory and, in any case, numerical results about chaotic systems are learned from pseudotrajectories not real trajectories.

The goal then is to produce a pseudotrajectory chosen from the uniform distribution over pseudotrajectories. A typical pseudotrajectory of length t can be generated using one processor in time linear in t by iterating the map using arithmetic of precision much better than δ and then adding a noise term on each step chosen from the uniform distribution on

$[-\delta, \delta]$. Could a typical trajectory be produced in far fewer than t parallel steps with the aid of many processor?

The model of parallel computation implicit in this question is the PRAM (parallel random access machine), the standard model in the theory of parallel computational complexity [2]. A PRAM is an idealized, fully scalable device with many identical (except for distinct integer labels) processors. Processor all run the same program and all communicate with a global memory in unit time. Massive parallelism is envisioned here however the number of processors is required to be polynomially bounded in $t \log(1/\delta)$, the effective number of (binary) degrees of freedom of a pseudotrajectory of length t and accuracy δ.

In the next section we show how to produce a typical trajectory in parallel using Monte Carlo path sampling. The procedure is correct but inefficient. We then describe how simulated annealing together with path sampling can produce a typical pseudotrajectory in parallel time that scales linearly in $\log t$ and polynomially in $\log(1/\delta)$ using a number of processors that is polynomial in $t \log(1/\delta)$.

II. PATH SAMPLING OF PSEUDOTRAJECTORIES

The method described in this section is based on path sampling ideas put forward by Chandler and collaborators [3]. The uniform probability density for pseudotrajectories , $\mathcal{P}(y_0, y_1, \ldots, y_t)$ is given by

$$\mathcal{P}(y_0, y_1, \ldots, y_t) = p(y_0) \prod_{n=1}^{t} P(y_n|y_{n-1}) \tag{3}$$

where

$$P(y'|y) = \begin{cases} \frac{1}{2\delta} & \text{if } |y' - f(y)| < \delta \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

and $p(y)$ is the invariant distribution.

A simple Monte Carlo procedure can be used to sample paths from $\mathcal{P}$. Given an existing pseudotrajectory, a single time $n > 0$ is chosen and a proposal for a new value for y_n is obtained according to

$$y'_n = f(y_{n-1}) + \epsilon \tag{5}$$

where ϵ is chosen from the uniform distribution on $[-\delta, \delta]$. The proposed value is accepted if it is also the case that $|y_{n+1} - f(y'_n)| < \delta$. It is straightfoward to verify that this Monte

Carlo procedure satisfies detailed balance with respect to $\mathcal{P}$. The question of ergodicity of the Markov chain in the space pseudotrajectories is less clear. However, even if ergodicity holds, the actual mixing time for the Monte Carlo procedure would be long when δ is small since the time to obtain an independent pseudotrajectory is at least as great as $1/\delta^2$, the time to diffuse a distance order one given a step size of δ. To be considered an efficient process for generating pseudotrajectories , the parallel time should increase no more rapidly than some power of the *logarithm* of $1/\delta$. This goal can be achieved by first using a simulated annealing procedure to produce a pseudotrajectory that is random on long time scales then to use the above Monte Carlo path sampling to randomize the small scales.

III. SIMULATED ANNEALING FOR PSEUDOTRAJECTORIES

Long pseudotrajectories are constructed by independently generating many *segments* or short trajectories and then "welding" the segments together. In the welding step the discontinuity between successive segments is eliminated by simulated annealing. In addition to simulated annealing, it is sometimes necessary to extend segments to obtain a weld to the next segment.

The fundamental time scale in the system is

$$\tau = -(\log \delta)/|\lambda|, \tag{6}$$

the time required for typical errors to grow to be order one. The length of the segments, K used in the construction should be longer than the fundamental time τ so that the beginning and end of each segment is uncorrelated. We also need to allow for extensions at either end of the segment and for a "warm-up" so that the initial point of the segment is chosen from invariant distribution. Thus, in practice, for each segment we start with a random number and iterate the map $L > K$ times, choosing the segment of length K from a predetermined part of the longer sequence of length L.

Having made a collection of segments, we now attempt to weld them together into a long pseudotrajectory . This is done in such a way that the initial value or *anchor point* of each segment is held fixed. The discontinuity between successive segments is annealed until all errors are less than δ. The Monte Carlo annealing procedure is designed to lower the error

$e(y_{n-1}, y_n, y_{n+1})$ associated with three successive elements, y_{n-1}, y_n and y_{n+1},

$$e(y_{n-1}, y_n, y_{n+1}) = |y_n - f(y_{n-1})| + |y_{n+1} - f(y_n)|. \tag{7}$$

If $e(y_{n-1}, y_n, y_{n+1}) < \delta$ for every n then we have a pseudotrajectory. For each time n, with the exception of the anchor points, the Monte Carlo annealing procedure begins with a measurement of $e = e(y_{n-1}, y_n, y_{n+1})$. If $e < \delta$ nothing is done. Otherwise a new value y'_n is proposed,

$$y'_n = y_n + \tilde{\epsilon} \tag{8}$$

where $\tilde{\epsilon}$ is chosen as a Gaussian random variable with mean zero and standard deviation $e/2$. If $e' = e(y_{n-1}, y'_n, y_{n+1})$, is less than e, the proposal is accepted as the new value for y_n. If the error increases, the proposal is accepted with probability $e^{-\beta(e'-e)}$. The value of the inverse temperature for each Monte Carlo step and is taken to be $\beta = e/2$ so that the acceptance ratio is independent of the size of the error. In a single Monte Carlo sweep, all except the initial and final values of each segment are processed using the above procedure. Given $t/2$ processors this can be done in constant parallel time by first processing the even and then the odd values of the time n.

In the chaotic regime ($\lambda > 0$) the annealing procedure should yield a valid pseudotrajectory for large enough K and sufficiently many Monte Carlo sweeps. In practice, however, some welds require a very large number of sweeps. Specifically, the probability distribution for the number of sweeps needed to achieve a weld has a long tail leading to parallel running times for creating pseudotrajectories that are dominated by the few most difficult welds. Two additional kinds of steps, called *forward shifts* and *backward shifts* cure this difficulty. Suppose segment s together with its final condition, the first element of segment $s + 1$, is not fully annealed after a predetermined number of annealing sweeps. Then segment s is restored to its original state and either it is extended forward one step or segment $s + 1$ is extended backwards one step. In the case of a backwards shift, the element prepended to segment $s + 1$ is considered a new anchor points and serves as the new final condition for segment s. The net effect of either a forward or backward shift is that the discontinuity between segments s and $s+1$ occurs with a different pair of numbers. Annealing sweeps and shifts are interleaved, a fixed number of Monte Carlo annealing sweeps are attempted and if all errors are not less than δ, a shift is done. The process is repeated until a satisfactory weld is achieved. Successive shifts are alternately of the forward and backward type. In Sec. V, I

show that the combination of Monte Carlo annealing and shifts produces a pseudotrajectory in $O(\log t)$ parallel steps.

Shifts serve several purpose. First, they simply provide for the possibility of more Monte Carlo sweeps, though if this were their only function it could be accomplished by directly increasing the number of sweeps. Second, shifts permit the algorithm to perform properly for periodic orbits or nearly periodic stretches of aperiodic orbits. The annealing procedure by itself cannot generate long pseudotrajectories for periodic orbits since welds are often attempted between segments that are out of phase with one another. Adding shifts to the annealing procedure insures that periodic pseudotrajectories will be correctly generated. For example, consider the case of a period two orbit, a single forward or backward shift of some segments will insure that all welds are satisfactory. For period d orbits, as many as $d-1$ shifts are necessary to insure that all welds are satisfactory.

Shifts may also provide padding around hard to weld regions of a trajectory. During a shift, new points are added to one end of a segment but no points are removed. Thus shifts do not bias the pseudotrajectory against difficult to weld regions in the invariant measure. For example, it is observed that if the final condition for a segment is very near the maximum of the support in the invariant measure at $r/4$ then one or more backward shifts are usually necessary so that the point near $r/4$ is surrounded by a region of small errors and is not involved in the annealing process.

IV.   FULL PARALLEL ALGORITHM FOR PSEUDOTRAJECTORIES

This section provides the details of the parallel algorithm for producing typical pseudotrajectories that combines the path sampling method of Sec. II and the annealing procedure of Sec. III. First the annealing procedure is used to generate a pseudotrajectory and then path sampling is used to further randomize it. The algorithm is controlled by several parameters: t is the total length of the desired pseudotrajectory , δ is the desired accuracy, K is the length of each segment, K' is the warm-up length, E, an even number, is the maximum number of shifts that are attempted, M_1 is the number of Monte Carlo annealing sweeps carried out between shifts and M_2 is the number of Monte Carlo path sampling sweeps. The algorithm is described below:

1. In parallel, generate $S = \lceil t/K \rceil$ sequences $\{x_m^{(s)}\}$ each of length $L = K' + K + E$,

with $s = 0, \cdots, S - 1$ and $m = 0, \cdots, L - 1$. The initial value of each sequence is a uniform random number on $(0, 1)$ and subsequent values are obtained by iterating the map $L - 1$ times to precision much greater than δ. This step requires $O(L)$ parallel time.

2. These S sequences are used to define ES segments $\{y_n^{(s,q)}\}$ each of length K, where the index q, $0 \leq q < E$ gives the number of shifts. $y_n^{(s,q)} = x_{n+E/2+K'+\lfloor q/2 \rfloor}^s$ for $0 \leq n < K - 1$ while $y_{K-1}^{(s,q)} = x_{E/2+K'-\lceil q/2 \rceil+K'}^{s+1}$. Note that the final point in segment $y^{(s,q)}$ is a taken from sequence $s + 1$.

3. In parallel, for each $s < S$, and each $q < E$, anneal segment $y^{(s,q)}$. The annealing procedure consists of M_1 Monte Carlo sweeps. During a single annealing sweep first the even and then the odd elements of the segment are updated in parallel. The initial and final points $y_0^{(s,q)}$ and $y_{K-1}^{(s,q)}$ are held fixed during the annealing procedure. A single Monte Carlo update of the point $y_m^{(s,q)}$ consists of the following procedure:

   - Compute $e = e(y_{m-1}^{(s,q)}, y_m^{(s,q)}, y_{m+1}^{(s,q)})$ from Eq. 7.
   - If $e < \delta$, do nothing. If $e > \delta$ propose a new value $y' = y_m^{(s,q)} + \tilde{\epsilon}$ where $\tilde{\epsilon}$ is a Gaussian random variable with mean zero and standard deviation $e/2$.
   - Compute $e' = e(y_{m-1}^{(s,q)}, y', y_{m+1}^{(s,q)})$. If $e' \leq e$ accept the proposed change, $y_m^{(s,q)} \leftarrow y'$. If $e' > e$ accept the proposed move with probability $\exp[-\beta(e'-e)]$ where $\beta = e/2$.

   The parallel time required for this step is $O(M_1)$.

4. In parallel, for each $s < S$ find $Q(s)$, that smallest value of q such that the segment is successfully annealed. The annealing is successful for this segment if, for all $0 < m < K - 1$, the errors are sufficiently small, $e(y_{m-1}^{(s,q)}, y_m^{(s,q)}, y_{m+1}^{(s,q)}) < \delta$. If for any s, annealing is unsuccessful for all $q \leq E$, the algorithm fails. This step can be carried out in constant parallel time.

5. The full pseudotrajectory y^* is a concatenation of sequences obtained from each original sequence $x^{(s)}$. The contribution to the pseudotrajectory from $x^{(s)}$ is the concatenation of $\{x_m^{(s)} | E/2 + K' - \lceil Q(s-1)/2 \rceil \leq m \leq E/2 + K' + \lfloor Q(s)/2 \rfloor\}$ and $\{y_m^{(s,Q(s))} | 0 < m < K - 1\}$. The first of these sequences is composed of the anchor points and the second sequence is the annealed segment. To obtain a pseudotrajectory

7

of length exactly t, the pseudotrajectory obtained above is simply truncated after t steps.

6. The path sampling Monte Carlo procedure described in Sec. II further randomizes y_m^*. During a single sweep, first all the even and then all the odd elements of the pseudotrajectory are updated in parallel. The number of sweeps is M_2. On each Monte Carlo step a new value for y_m^* is proposed according to Eq. 5 and accepted only if the trajectory is still a pseudotrajectory within error δ. The randomization step requires parallel time $O(M_2)$.

V.   VALIDITY AND COMPLEXITY OF THE PARALLEL ALGORITHM

The central questions addressed in this section are (1) whether the algorithm succeeds in creating a pseudotrajectory, (2) how the scaling of the number of parallel steps depends on the length and accuracy of the pseudotrajectory and the parameter r of the map and (3) whether the algorithm samples the uniform distribution on pseudotrajectories. A sequential algorithm that carries out the annealing and path sampling routines one segment at a time was used to study these questions. In the simulations reported below, the parameters are chosen to be $M_1 = M_2 = 5\tau^2$, $K = 5\tau$ and $K' = 1000$. The assumption behind these choices is that memory is lost on a time scale τ so that placing independently chosen anchor points separated by $K = 5\tau$ is satisfactory. The annealing process that welds successive segments is expected to influence a region whose length is order τ. Since information is transmitted diffusively by local Monte Carlo moves, having the number of Monte Carlo sweeps scale as τ^2 should suffice.

The annealing stage of the parallel algorithm can be studied one segment at a time since each segment is independently annealed. First, I observed that, given enough shifts, the annealing step always produced a successful weld. The choice of the maximum number of shifts E for the annealing stage must be large enough to make the failure probability for the whole algorithm small. For long trajectories, $t \gg \tau$, the choice of E is determined by the tail of the distribution of the number of shifts, Q required to obtain a weld since the whole procedure fails if even one segment is not successfully annealed. Suppose $C(\cdot)$ is the cumulative probability distribution for Q. An estimate of the maximum number of shifts,
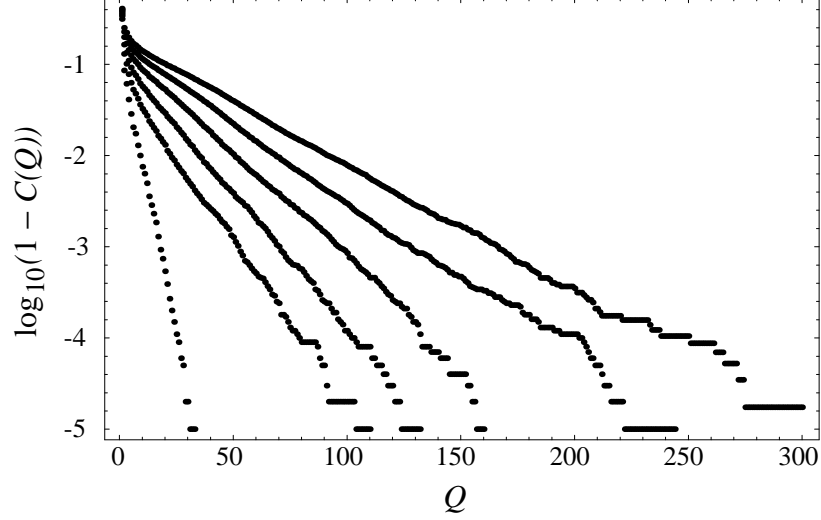
FIG. 1: Logarithm of the complement of the cumulative distribution for the number of shifts, $\log_{10}(1 - C(Q))$ vs. number of shifts, Q. From left to right, $\delta = 10^{-5}$ to 10^{-10}.

E needed to insure all segments are annealed is given by the relation

$$(t/K)(1 - C(E)) < 1. \tag{9}$$

Figure 1 shows $\log_{10}(1 - C(Q))$ vs. Q. These data were collected for the case of $r = 3.7$ (the period doubling transition to chaos occurs at $r = 3.5699\ldots$) and the six curves from left to right are for $\delta = 10^{-5}$ through 10^{-10}, respectively. Each curve is obtained from annealing 10^5 segments except for the $\delta = 10^{-10}$ curve which is obtained from 6×10^4 segments. For $r = 3.7$ the Lyapunov exponent is $\lambda = 0.354$ and so, for example, with $\delta = 10^{-7}$, $\tau = 45.5$, $K = 228$ and $M_1 = 10366$. Over a reasonable range following an initial transient and before the noise becomes large, the data falls on straight lines suggesting that the distribution is asymptotically exponential, $C(Q) \sim 1 - \exp(-Q/\sigma)$. Equation 9 then implies that $E \sim \sigma \ln t$ and we can conclude that the parallel running time is $O(\log t)$ since no other contribution to the running time depends on the overall length of the pseudotrajectory.

How does the decay constant σ and thus the running time depend on the choice of the accuracy δ. Figure 2 shows σ vs. δ on a logarithmic scale for the case $r = 3.7$ and suggests that σ is a polynomial function of $\log \delta$. The other simulation parameters, L, M_1 and M_2 are also polynomial in $\log \delta$ so we conclude that the full algorithm has a running time that is polynomial in $\log \delta$ and linear in $\log t$.

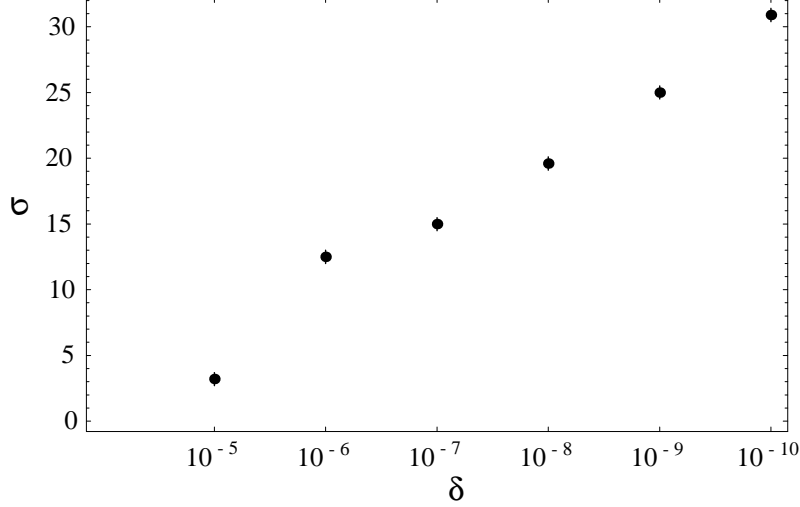I also considered two other parameter values for the quadratic map, $r = 3.6$ where $\lambda =$

FIG. 2: Decay constant σ for the distribution of shifts vs. accuracy δ on a logarithmic scale.

0.183 and $r = 3.95$ where $\lambda = 0.577$. In both cases, the accuracy was set to $\delta = 10^{-7}$. The decay of $(1 - C(Q))$ appears to be exponential in both cases but with rather different values of the decay constant: $\sigma = 450, 15$ and 9.5 for $r = 3.6, 3.7$ and 3.95, respectively. Either σ depends strongly on λ or perhaps there are additional r dependent factors controlling σ. For example, for $r = 3.7$ and 3.8 the invariant measure has support on a single interval but for $r = 3.6$ the support consists of two intervals.

The annealing stage of the algorithm creates a pseudotrajectory but it is not typical in the sense of being chosen from the distribution of Eq. 3. On long time scales, the pseudotrajectory is randomized by the random choice of initial conditions for each sequence. However, the individual errors, $\epsilon_n = y_{n+1}^* - f(y_n^*)$ are not guaranteed to be independent random variables on the interval $[-\delta, \delta]$. For example, anchor regions of the pseudotrajectory have errors much less than δ. The hypothesis is that $M_2 = O(\tau^2)$ path sampling Monte Carlo sweeps are sufficient to randomize the short time scales and produce a typical pseudotrajectory from the pseudotrajectory produced by the annealing stage. To check this hypothesis, I computed mean values and autocorrelation functions for errors and cross correlations between errors and values of the pseudotrajectory . The quantities $\langle \epsilon_n \rangle$, $(\langle \epsilon_n^2 \rangle - \delta^2/3)$, $\langle \epsilon_{n+1} \epsilon_n \rangle$, $\langle \epsilon_{n+1} y_n^* \rangle$ and $(\langle \epsilon_{n+1}^2 \epsilon_n^2 \rangle - \delta^4/9)$ were all found to be zero within error bars for the case $r = 3.7$ and $\delta = 10^{-7}$. Here the angled brackets indicate an average over segments and over n. The vanishing of these quantities is a necessary but not sufficient condition that pseudotrajectory is chosen from the uniform distribution described by Eqs. 3 and 4. More work is needed to

10

firmly establish that the algorithm with $O(\tau^2)$ path sampling sweeps samples the uniform distribution to good approximation. However, even without the path sampling stage, the pseudotrajectories produced by the annealing stage are typical in a different sense. As shown in [1], any pseudotrajectory *shadows* an exact trajectory (i. e. remains close to over its entire length) though possibly for a larger value of r.

VI.   CONCLUSIONS

I have exhibited a parallel algorithm that generates pseudotrajectories of the quadratic map. Numerical evidence suggests that the parallel time required to generate a typical pseudotrajectory increases linearly in $\log t$ and polynomially in $\log(1/\delta)$ though more work would be required to establish these scalings with certainty. Essentially the same parallel algorithm can be applied to other one-dimensional and higher dimensional maps. It would be interesting to explore whether the annealing/shift procedure is sufficient to efficiently sample pseudotrajectories for other maps.

Since there is little demand for very long pseudotrajectories of the quadratic map, the fast parallel algorithm is probably not of practical value. The significance of its existence and complexity is that it characterizes the history dependence of the map. The existence of a fast parallel simulation is a strong statement against history dependence since it shows that the logical path from independent random numbers (used to drive the Monte Carlo procedures) to a typical pseudotrajectory is much shorter than the length of the pseudotrajectory. The length of this logical path is one measure of the potential for generating historical complexity. In very few logical steps, very little complexity can be arise. The idea that complexity tends to emerge at the "edge of chaos" [4] is born out here since the basic time scale τ for the parallel algorithm diverges when the Lyapunov exponent vanishes. An appealing feature of characterizing systems by computational complexity is that very different systems in statistical physics systems, for example diffusion limited aggregation [5], sandpiles [6] or the Bak-Sneppen model [7], can be compared to one another within the same framework.

---

[1] H. E. Nusse and J. A. Yorke. Is every approximate trajectory of some process near an exact trajectory of a nearby process? *Commun. Math. Phys.*, 114:363–379, 1988.

[2] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to Parallel Computation: P-completeness Theory.* Oxford University Press, 1995.

[3] C Dellago, P. G. Bolhuis, F. S. Csajka, and D. Chandler. Transition path sampling and the calculation of rate constants. *J. Chem. Phys.*, 108:1964–1977, 1998.

[4] C. G. Langton. Computation at the edge of chaos. *Physica D*, 42:12–37, 1990.

[5] K. Moriarty, J. Machta, and R. Greenlaw. Parallel algorithm and dynamic exponent for diffusion-limited aggregation. *Phys. Rev. E*, 55:6211, 1997.

[6] C. Moore and M. Nilsson. The computational complexity of sandpiles. *J. Stat. Phys.*, 96:205–224, 1999.

[7] J. Machta and X.-N. Li. Parallel dynamics and computational complexity of the Bak-Sneppen model. *Physica A*, 300:245–270, 2001.